Improving Rule Bases for Agent-Based Simulations

Todd Graves, Rick Picard, and Stephen Upton

Abstract

In agent-based simulation codes, where rule bases guide agents through a process, discovering desirable rule sets is crucial. In the motivating example for this report, "agents" could represent soldiers in a simulated military conflict and a rule base defines when individual agents take various actions (e.g., advancing towards enemy lines or shooting at enemy soldiers). We develop a sequential approach for identifying rule bases that provide desirable outcomes, as well as giving an understanding of which rules are most important. By working in the space of contingency-table-like odds ratios instead of in the space where the rule base is directly defined (e.g., as a bit string with a finite alphabet), the fitness function behaves more smoothly and optimization is more easily pursued. The computational approach also nicely complements results from subjective visualization procedures.

Acknowledgment

This work was supported as part of Project Albert at the United States Marine Corps Combat Development Command, Quantico, VA.

1 Introduction

Increased computing power in recent years has allowed for computer simulation of increasingly detailed processes. Agent-based simulation involves individual "agents" which interact with each other and carry out various functions. In the example considered later, each agent is a soldier fighting in a simulated conflict. Other examples of agent-based simulations include such wide ranging applications as vehicles moving in a transportation network (Nagel, Rickert, and Barrett 1997), biological organisms evolving in the presence of threats (Jaffe et. al. 1997), individual investors behaving as part of a collective financial system (Deadman 1999), and products being assembled in an industrial manufacturing process (Fogerty and Bull 1995).

In such simulations, each agent consults a rule base that defines what actions take place at each time step. For the military example to follow, soldiers may move, adopt defensive postures, shoot at enemy soldiers, and so on. Of interest are properties of the system's emergent behavior, such as the probability that a particular army wins the battle.

To provide realism, agent-based codes are often stochastic. In military simulations, simple examples of stochastic elements are line-of-sight probabilities (i.e., the chance that one soldier will see another who's in the vicinity) and kill probabilities (i.e., the probability that one soldier will kill another, given that he sees him and is shooting at him). More generally, stochastic elements in agent-based simulations can involve initial conditions, agent travel time distributions, random system breakdowns and times to repair, queuing phenomena, or even agents' choices of actions. Owing to such stochastic behavior, statistical inference is essential.

Frequently, it is of interest to identify the best rule base governing agent behavior. Depending on the simulation involved, the term "best" could mean finding the rule base that gives an army its best chance to win a simulated battle, or whose simulated traffic is in best agreement with observed data, or whose simulated widget production is the greatest. In this report, we express rule bases as fixed-length strings with finite alphabets (tree structures could alternatively be used), so that standard optimization techniques can be pursued, such as genetic algorithms and simulated annealing. As will be seen, however, rule base fitness behaves irregularly as a function of bit strings, which adversely affects the optimization process. An additional complication is that fitness is affected by factors other than the rule base, such as the nature of the competing forces together with their arms and other equipment.

Moreover, it is not enough to obtain a "black box" solution providing little intuition regarding which rules matter and which ones don't. For purposes of using simulation results to support training of actual soldiers, it is important to identify specific rules that greatly affect performance. Because it is impractical to effectively teach an entire rule base covering all possible situations in which an agent could find himself, extracting important if-then relationships is essential. The approach described here achieves that goal.

The approach discussed here also differs from visualization in that results are scientifically reproducible. Although visualization allows the user to see relationships that might be difficult to discern otherwise, this strength can also be a serious weakness. In the extreme, visualization can be similar to psychology's Rorschach test, where people give highly personalized impressions as to what they "see" in ink blots.

In this paper, we describe a sequential method to extract useful rule bases for agent-based simulations, and apply it to a simplified problem. The approach is regression-based, can be implemented with standard statistical software, and yields results that nicely complement those from visualization or from optimization techniques such as genetic algorithms.

2 Combat Simulation

2.1 Background

To focus ideas, consider a simplified armed conflict between the good guys (more formally, "the blue team") and the bad guys ("the red team"). Given a set of initial conditions, simulations are performed to develop understanding of the conflict's dynamics. There are many such war gaming codes (see, e.g., Ilachinski 1997, as well as the TRADOC and CACI reports given in the reference list).

In agent-based simulation, it is common to simulate action during discrete time steps. At each time step, each agent evaluates his status, uses a rule base to determine what actions to take, and the code then synchronizes the actions for all agents during the time step. Usually, simulated behavior is stochastic – e.g., if one soldier shoots at another, the outcome is computed using a specified probability distribution that incorporates factors such as the distance to the other agent and whether that agent is in defensive posture.

Of interest is the so-called emergent behavior of the system, which arises upon evolving the collective actions of the agents over the simulated time steps. Emergent behavior is a function not only of rule base that guides agents' actions, but also of the initial conditions for the conflict and of how the soldiers are equipped (e.g., their weapons, vehicles, sensors and properties thereof). In what follows we focus exclusively on the rule base, fixing the initial conditions and equipment. The goal is to find rule bases that maximize, to the extent possible, blue team performance.

At present, military analysts often carry out a heuristic tweaking of rule bases by varying one factor at a time, trying to identify important actions. In addition, available optimization procedures (e.g., genetic algorithms) are sometimes used to provide solutions. Though helpful, the output of such algorithms frequently has a black box quality, providing an optimum but little intuition about the importance of individual rules.

2.2 JIVES Town

We consider a simulated urban conflict in a notional city called JIVES Town. The acronym JIVES (Joint Integrated Virtual Environment for Simulation) refers to a specific combat simulation code written in the Military Systems Group at Los Alamos (Upton et. al. 1998). This code is well suited to handle input/output for multiple rule base runs and, more importantly, it is research friendly in that it allows great flexibility in bookkeeping. Thus, the user can track aspects of immediate interest, without being inundated with unwanted output. This feature is valuable with respect to calculating the odds ratios discussed later.

A terrorist-scale conflict is examined, where 8 blue agents square off against 4 red agents. At time zero, the blue agents start at locations specified depicted in Figure 1 with orders to advance towards city hall. They have 100 time steps to get there. The red agents are in a defensive mode, stationed along a side street to prevent blue from advancing. They have a fixed rule base, remaining in their initial locations, always in a defensive posture, always looking, and shooting whenever they sense a blue agent. The blue agents' rule base is to be determined: blue agents are allowed to move, look, shoot, change their posture, and if they are both shooting and changing posture, to invert the sequential order in which these actions are carried out.

The status of each blue agent at each point in time is described by five binary variables:

(a) whether the agent senses one or more red agents in the immediate vicinity or not,

(b) whether the agent senses one or more blue agents in the immediate vicinity or not,

(c) whether the agent is under fire or not,

(d) the agent's current posture (offensive or defensive), and

(e) whether the agent has reached his goal, or not.

Regarding (d), the agent's posture, each agent may choose to be in offensive posture (which allows movement of a greater distance per time step) or defensive posture (which reduces vulnerability to enemy fire). Regarding (e), an agent is defined to be at his goal once he is within a specified distance of a goal location.

The possible (binary) actions that each blue agent can take at each time step are

(a) if sense one or more red agents in the immediate vicinity, shoot at one of them, or not,

(b) change posture (e.g., from offensive to defensive), or not,

(c) if shooting and changing posture, decide in which sequential order in the same time step these actions take place,

(d) move towards the goal at a posture-dependent rate, or not, and

(e) look around the immediate vicinity (checking for presence of other agents to be sensed in the next time step), or not.

It is desired to find the rule base which gives the best average performance for blue team. That is, to determine under what circumstances blue agents should look, move, adopt defensive postures, and so on. The rule base for the red team is fixed throughout, avoiding for now problematic game-theoretic issues where the red team modifies its own rule base to counter changes in the blue team's tactics.

The human analyst specified, a priori, a rule base that seemed to him to be optimal based on experience with other simulation codes. That rule

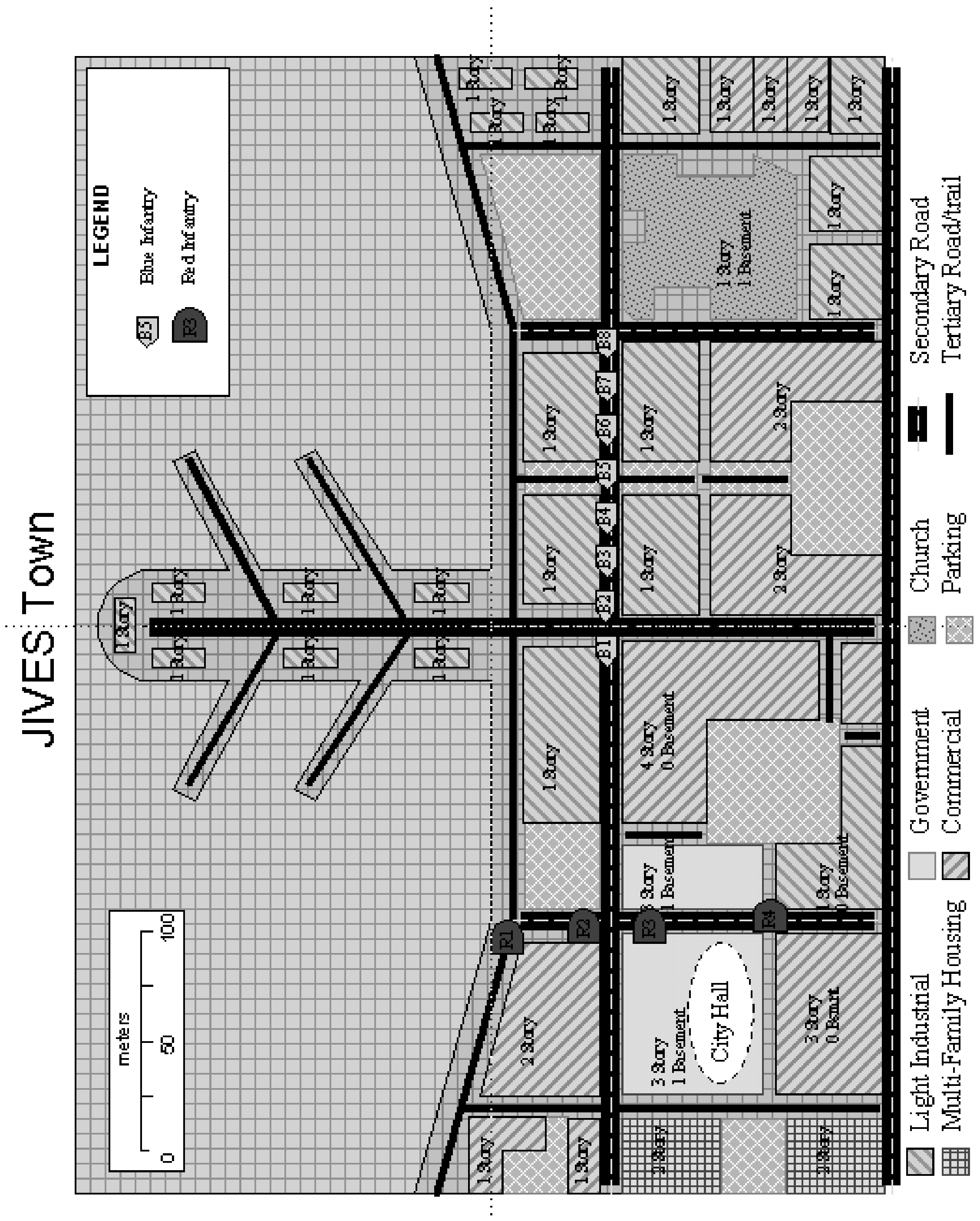

Figure 1: JivesTown

base is displayed in Table 1 and specifies for each blue agent to always look, always move, to shoot if in defensive posture and sense a red agent nearby, to adopt defensive posture and then shoot if in offensive posture and sense a red agent nearby, to adopt a defensive posture if under fire but don't sense a red agent nearby, and to adopt offensive posture if in defensive posture and don't sense a red agent nearby and are not under fire. Our goal is to find better rule bases, if possible, and to understand the importance of taking various actions under various conditions.

3 Analysis of Simulation Results

3.1 Modeling the Data

In the approach discussed below, we use regression modeling to aid in discovering good rule bases. Because emergent behavior is complex, there is no first-principles theory for it and the regression model is an empirical one, intended to identify important if-then rules.

We take as the response variable the average number of blue agents who reach the goal and are alive at the end of the battle. In a sense, this fitness function is equivalent to the probability of a single blue agent's success (just divide by 8). An initial tendency is to model the response as a function of covariates. As is apparent from Table 1, a rule base is completely defined by a string of 160 bits, detailing which of the 5 actions are taken under each of the 32 possible conditions an agent might face in a given time step.

Models based on the related bit string could be considered. Upon allowing model parameters for combinations of the 160 bits to explore effects of actions taken 2 and 3 at a time, such models contain literally thousands of terms. If it were practical to obtain results from millions of rule bases, models of this form could be pursued, together with the dimension reduction techniques required to distill the results into something interpretable.

Such an approach is tractable in the JIVES Town setting when there is access to high performance computing, but not in more realistic problems where the number of status/action variables is much larger than the 10 at present, where such variables may be polytomous (vs. binary) or continuous, and where different subsets of agents can have their own respective rule bases. Thus, in more realistic simulations there are far more rule bases than the roughly 10^{35} in the current problem (meaning that the space to be

Status Variables					Action Variables				
SeeRed	SeeBlue	UnderFire	Posture	AtGoal	Shoot	Move	ΔPosture	Look	ΔOrder
0	0	0	0	0	0	1	1	1	0
1	0	0	0	0	1	1	0	1	0
0	1	0	0	0	0	1	1	1	0
1	1	0	0	0	1	1	0	1	0
0	0	1	0	0	0	1	0	1	0
1	0	1	0	0	1	1	0	1	0
0	1	1	0	0	0	1	0	1	0
1	1	1	0	0	1	1	0	1	0
0	0	0	1	0	0	1	0	1	0
1	0	0	1	0	1	1	1	1	1
0	1	0	1	0	0	1	0	1	0
1	1	0	1	0	1	1	1	1	1
0	0	1	1	0	0	1	1	1	0
1	0	1	1	0	1	1	1	1	1
0	1	1	1	0	0	1	1	1	0
1	1	1	1	0	1	1	1	1	1
0	0	0	0	1	0	1	1	1	0
1	0	0	0	1	1	1	0	1	0
0	1	0	0	1	0	1	1	1	0
1	1	0	0	1	1	1	0	1	0
0	0	1	0	1	0	1	0	1	0
1	0	1	0	1	1	1	0	1	0
0	1	1	0	1	0	1	0	1	0
1	1	1	0	1	1	1	0	1	0
0	0	0	1	1	0	1	0	1	0
1	0	0	1	1	1	1	1	1	1
0	1	0	1	1	0	1	0	1	0
1	1	0	1	1	1	1	1	1	1
0	0	1	1	1	0	1	1	1	0
1	0	1	1	1	1	1	1	1	1
0	1	1	1	1	0	1	1	1	0
1	1	1	1	1	1	1	1	1	1

Table 1: The rule base supplied by the expert analyst. The first five columns are status variables, and the next five columns describe the actions that blue agents take when encountering the conditions given by the five status variables. A ‘1’ in the table denotes a ‘yes’ and a ‘0’ denotes a ‘no.’

searched is much larger), and there is the double whammy that the additional simulation complexity means that a single simulated battle requires far more CPU time than for a single JIVES Town run. Consequently, making millions upon millions of runs is generally impractical.

With an eye towards making progress in more complex settings, we choose to model the fitness function in terms of other output responses instead of modeling it in terms of rule base descriptors such as bit values in a bit string. The response variables we choose are more meaningful and their space has much smaller dimension than for combinations of bits. In addition, the fitness function behaves more smoothly as related to those responses than with respect to bits in a bit string, where altering a single bit can greatly affect fitness.

Regressor variables to follow are formed by using log odds ratios of a rule base's empirical behavior. For the i^{th} (of 5) action variables in Table 1, define its "main effect" as the odds ratio

$$\alpha_i = \frac{(0.5 + \#\text{times take action } i)/(1 + \#\text{times could have taken action } i)}{(0.5 + \#\text{times didn't take action } i)\ /(1 + \#\text{times could have taken action } i)}\ .$$

If all 8 blue agents were to remain alive for all 100 time steps in all r replicated runs for the rule base, then the number of times that action i could have been taken would be $8 \times 100 \times r = 800r$. Main effects for the 5 status variables are defined similarly, counting the number of times that each condition occurred and didn't occur. The factors 0.5 and 1 in the odds ratio expression for α_i are included to avoid numerical problems with zero counts.

Regressor variables are next constructed for the "two-factor interactions" between variables. For action variable i and status variable j, for example, this interaction is summarized by the odds ratio

$$\beta_{ij} = \frac{(0.5 + \#\text{times take action } i \text{ when in status } j)/(1 + \#\text{times in status } j)}{(0.5 + \#\text{times don't take action } i \text{ when in status } j)/(1 + \#\text{times in status } j)}$$
$$\times \frac{(0.5 + \#\text{times don't take action } i \text{ when not in status } j)/(1 + \#\text{times not in status } j)}{(0.5 + \#\text{times take action } i \text{ when not in status } j)/(1 + \#\text{times not in status } j)}\ .$$

The term β_{ij} is related to conditional independence structures in contingency tables (see, e.g., Bishop, Fienberg, and Holland 1975, p. 13). That is, consider the 2×2 table having row labels "take action i" and "not take action i" and column labels "in status j" and "not in status j". Cell counts for the table denote the total number of times during the simulation of the rule base that blue agents were in the status indicated and acted as described. From these counts β_{ij} is computed. Were the log of β_{ij} to equal zero, for example, there would no relative preference, in an overall sense, for agents to take action i more/less often when in status j than when not in status j.

The odds ratios $\{\alpha_i\}$, $\{\beta_{ij}\}$, and their counterparts for higher-order interactions summarize, in an aggregate sense, the behavior of agents who adhere to the rule base. More thoroughly, odds ratios could be extended to summarize sequences of actions covering 2 or 3 time steps, in an attempt to understand complex behavior and to account for agents having a memory of past events.

[Note. Odds ratios from rule bases are not interpretable in exactly the same way as for contingency tables. The cell counts in a table of rule firings don't evolve from an independent, identically distributed (i.i.d.) sampling process. Also, some care in interpretation of these odds ratios is occasionally required owing to Simpson's paradox issues (Samuels 1993) for collapsed tables. Nonetheless, we have found odds ratios to be valuable in model fitting.]

Our approach is to regress the average response for each rule base on the set of log odds ratios. Questions such as "do rule bases whose agents move frequently perform better than rule bases whose agents don't?" can then be directly addressed by examining parameter estimates from the fit. And higher-order parameters also have interpretable meaning. For example, suppose that the estimated coefficient of the log odds ratio β_{ij} for the action variable "move" and status variable "under fire" were negative and deemed significant. This would indicate that rule bases performed better when their agents moved less often when under fire than when not under fire. Although such information doesn't completely determine under which conditions agents should move, it helps define a good set of rule bases for future consideration.

3.2 Searching Rule Base Space

Search techniques should be tailored for the problems to which they are applied. Important aspects of the surface to be optimized are the following.

(a) Expressing the blue team rule base in the form of Table 1, there are 32 possible status conditions at each time step and 8 to 32 possible actions for each condition (recall the constraints that an agent must sense an enemy in order to shoot, and must be shooting and changing posture in order to choose the sequential order in which these two actions take place). Thus, the space of allowable rule bases contains roughly 10^{35} members, precluding anything approaching an exhaustive simulation of all allowable rule bases.

(b) The code is stochastic, so that multiple simulation runs are needed to estimate the average performance of an individual rule base.

(c) The vast majority of allowable rule bases give terrible performance, with few if any blue agents ever reaching the goal.

(d) The surface is highly irregular when expressed as a function of bit string representation; e.g., changing a single action for a single status in a rule base can greatly alter average performance.

The bottom line is that it is impossible to guarantee that a global optimum will be found through *any* optimization procedure. A realistic goal is to understand the if-then relationships important to good performance.

Despite the above complications, the problem is well suited to genetic algorithms, which are adept at searching spaces of bit strings. This approach has been pursued with the ISAAC code (Ilachinski 1997), but the black box nature of the solution often doesn't help in understanding which rules are important and which aren't.

In order to generate data for regression modeling, it is necessary to simulate output from several rule bases. An obvious question is how to choose those rule bases. In what follows, we take a sequential approach, common to statistical applications of experimental design in other settings. That is, we start with an initial set of rule bases, analyze the results, choose another set of rule bases for further investigation, and so on.

For the first iteration of this sequential process, we choose a space-filling design. Many approaches to filling bit string space exist. In what follows, we construct random rule bases by letting each of the 160 bits be one or zero independently with probability 1/2 each. We thus obtain 640 rule bases, and run each of them ten times. This requires less than 24 hours of computing time on a PC. At this early stage of the sequential design, a few runs per rule base are sufficient to distinguish very bad rule bases from more promising ones. In later iterations of the sequential design, we require more runs in order to detect more subtle differences, because variability increases with the average response and the rule bases of greatest interest have the largest variability.

This initial set of 640 rule bases falls far short of filling the space of roughly 10^{35} possible rule bases very densely, but it nevertheless gives some useful results. An artifact of space filling designs in this context is that the vast majority of rule bases chosen randomly gave poor blue team performance: in our design, 557 out of 640 (87%) fail to have any blue agents reach their goal, and only 24 rule bases (4%) average at least one agent per run achieving the goal. This poor performance is not necessarily bad, in that it's helpful to learn what rules an agent should *not* follow. When combined with data from good rule bases run in later iterations of the sequential design, the data from poor rule bases aids in parameter estimation. Interestingly, the best of the 640 randomly chosen rule bases yields an average of 3.6 blue agents at the goal, which is very close to the performance of the rule base constructed by the expert analyst.

In early iterations of the design, parameter estimation is highly variable. Because most of the rule bases are poor ones, the model fitting process is similar to fitting outliers in the presence of multicollinearity. Certain estimated coefficients in the model have the wrong sign. For example, the coefficient for the action "Look" as estimated from the randomly generated rule bases is negative, which gives the (incorrect) impression that it is better for agents not to look. The cause of this problem is the multicollinearity between the "Look" and "Shoot" actions (i.e., because it is necessary to look before shooting, rule bases whose agents rarely look also rarely shoot, those whose agents shoot often also look often. Thus, the empirical analysis has difficulty separating the effects of looking from the effects of shooting).

Upon inspection of the results from the first iteration, a second group of 384 (or 6×64) rule bases is run, with 100 replications of each rule base. For each of the 6 groups, certain actions are fixed based on the most important regression coefficients and the rest of the rule base is filled out at random. The 6 groups of rule bases examined are:

(1) Always shoot when sense red agents nearby.

(2) Never look.

(3) Always be in defensive posture.

(4) Move if a red agent is sensed nearby.

(5) Don't change sequential order (of shooting and changing posture) when under fire.

(6) Shoot when sense a red agent nearby and a blue agent is not sensed nearby.

Results from these runs are combined with those from the 640 randomly chosen rule bases and parameters in the regression model are re-estimated.

These results indicate promising pairs of actions to consider. Six more groups are then considered, the corresponding pairs of actions being

(1) Always move and shoot when sense a red agent nearby.

(2) Always look and shoot when sense a red agent nearby.

(3) Always be in defensive posture and shoot when sense a red agent nearby.

(4) Always move and be in defensive posture.

(5) Always look and be in defensive posture.

(6) Always move and look.

As in the first iteration, bit values not fixed are selected at random, there being 64 such rule bases generated for each of the 6 groups. The ensuing data are then combined with earlier results and model parameter estimates are updated.

The sequential process continues, using model results to determine new rule bases to consider, simulating according to those rule bases, combining the results with those obtained previously, updating parameter estimates, and so on.

We chose the rules to fix in each group of rule bases by examining estimated regression coefficients. Consider Table 2, which contains parameter estimates for a fitted model to all of the data from all of the rule bases that were simulated throughout the sequential design. Of the main effects for the action variables, the largest coefficients are for Move (+0.14) and Look (+0.09), indicating that rule bases whose agents moved and looked most frequently performed better than rule bases whose agents didn't. The largest status-by-action variable interaction is for SeeRed $\times$ Shoot (+0.22), indicating that rule bases whose agents shoot when the see red agents nearby do better than rule bases whose agents don't.

Such coefficient values lend themselves to candidate rule bases. For example, the large coefficient for the Move main effect could generate a candidate rule base whose agents move always and who take other actions (e.g., when to change posture) in a manner defined randomly. Two-factor interactions lead to similar candidate rule bases, e.g., agents should always shoot when a red agent is sensed nearby and should take other actions in a manner defined randomly. In the analysis here, decisions as to how many actions to hold constant per iteration of the sequential design are made in a subjective manner; perhaps a more formal algorithm for this could be developed in the future.

More thorough interpretation of the estimated regression coefficients in Table 2 is given in the next section.

3.3 Estimated Regression Model

After four iterations of rule base designs, the sequential process was (arbitrarily) terminated. The regression model as fit to all of the data is summarized in Table 2. Its R^2 value, 0.83, is indicative of reasonable predictive power.

Candidate odds ratios for the model involved 250 terms. There were 5 odds ratios summarizing main effects of action variables, 5 more for main effects of status variables, 10 for two-factor action-by-action interactions, 10 for two-factor status-by-status interactions, 25 for action-by-status interactions, 50 for three-factor action-by-status-by-status interactions, 50 for three-factor action-by-action-by-status interactions, and 100 for four-factor action-by-action-by-status-by-status interactions. Models of this form can be easily fit using standard statistical software (in our case, the S-plus package). Estimates were obtained using ordinary least squares, which are relatively efficient in light of the moderate heteroscedasticity present.

Upon examining the candidate odds ratios, terms which appeared to add little to the fit were dropped from consideration. By the end of the fourth iteration of the design, the final model incorporated 38 terms, as described in Table 2.

By examining the estimated model coefficients in the final model, important if-then relationships are revealed. Of all the model coefficients involving action variables, the largest in magnitude (+0.22) corresponds to the shoot-

Variable	coef	std.err.	t stat	p value
Intercept	-0.18	0.18	-0.98	0.33
SeeRed	0.71	0.12	6.17	<0.001
SeeBlue	-0.51	0.12	-4.44	<0.001
UnderFire	-0.22	0.03	-6.98	<0.001
Posture	-0.13	0.01	-11.43	<0.001
Shoot	0.03	0.03	1.07	0.28
Move	0.14	0.02	8.02	<0.001
ΔPosture	0.00	0.004	0.02	0.99
Look	0.09	0.02	4.1	<0.001
ΔOrder	0.00	0.003	0.08	0.93
SeeRed × SeeBlue	-0.25	0.02	-16.8	<0.001
SeeRed × UnderFire	0.31	0.10	3.11	0.002
SeeRed × Posture	-0.09	0.01	-8.72	<0.001
SeeBlue × UnderFire	-0.27	0.09	-2.91	0.004
SeeBlue × Posture	0.08	0.01	6.71	<0.001
UnderFire × Posture	-0.09	0.01	-6.87	<0.001
Shoot × Move	0.09	0.02	5.13	<0.001
Move × Look	0.01	0.003	2.94	0.003
SeeRed × Shoot	0.22	0.02	13.29	<0.001
SeeRed × Move	0.04	0.01	3.24	0.001
SeeBlue × Move	-0.07	0.01	-6.73	<0.001
SeeBlue × Look	-0.06	0.01	-6.38	<0.001
UnderFire × Move	-0.04	0.006	-6.69	<0.001
Post. × Look	-0.02	0.004	-4.87	<0.001
SeeRed × SeeBlue × Shoot	-0.2	0.01	-12.74	<0.001
SeeRed × SeeBlue × Look	0.03	0.009	3.47	0.001
SeeRed × Posture × Shoot	-0.05	0.007	-8.19	<0.0010
SeeRed × Posture × Move	-0.04	0.006	-7.35	<0.001
SeeBlue × Posture × Shoot	0.06	0.009	6.36	<0.001
SeeBlue × Posture × Move	0.03	0.006	4.94	<0.001
SeeRed × Shoot × Move	0.03	0.008	3.04	0.002
SeeBlue × Shoot × Move	-0.05	0.009	-5.62	<0.001
SeeBlue × Shoot × Look	-0.05	0.01	-4.93	<0.001
UnderFire × Shoot × Move	-0.01	0.004	-3.87	<0.001
Posture × Shoot × Look	-0.01	0.003	-3.86	<0.001
Posture × Move × Look	-0.006	0.002	-3.35	0.001
SeeRed × SeeBlue × Shoot × Look	0.03	0.006	5.62	<0.001
SeeRed × Posture × Shoot × Move	-0.04	0.004	-8.48	<0.001
SeeBlue × Posture × Shoot × Move	0.02	0.005	4.25	<0.001

Table 2: Estimated regression model.

by-sense-red interaction. Stated simply, blue agents who shoot when sensing red agents nearby do better than blue agents who don't. Similarly, the main effects for moving (+0.14) and looking (+0.09) are positive and significant, indicating that agents should be moving and looking a large portion of the time (if not doing so all of the time). These conclusions aren't earthshaking, but the point is that a purely empirical review of simulation results has produced them.

Regarding one of the status variables, note that the coefficient for posture (-0.13) is significantly negative. When coupled with an insignificant "change posture" main effect and and absence of interactions with a "change posture" component, this indicates that rule bases whose agents remained in defensive posture a large portion of the time performed better than rule bases whose agents didn't. Indeed the best observed rule base (Table 3 of the next section) had this quality.

Interestingly, agent posture is one area where the modeling approach yielded a different solution than was postulated initially in the expert rule base. Recall that the expert rule base involved agents being in defensive posture only when encountering red agents or under fire. The number of time steps involved in these simulations, 100, is sufficiently large that agents can be in defensive posture all of the time, move most of the time, and still reach the goal. Thus, the fitness function does not reward the increased rate of movement offered by offensive posture, while it penalizes the increased vulnerability to enemy fire. Were a smaller number of time steps to be involved or a different metric used to define mission effectiveness, this result might change.

Another area where the modeling approach appears to differ with the expert rule base is related to the Move$\times$UnderFire interaction term (-0.04). This indicates that there may be some benefit to agents who don't move some of the time when under fire, perhaps allowing trailing blue agents who are not under fire to catch up with them, thereby creating a numerical advantage when encountering red agents. While this result doesn't specify exactly when agents should/shouldn't move, it indicates a behavior worth exploring. As it turns out, the agents in the best rule base we encountered did not move when under fire.

One point of this analysis is to show that using modeling and a purely empirical approach, it is possible to venture into parts of the rule base space that might not be investigated otherwise. Because the use of expert opinion will (and should) be used as part of an overall analysis, the use of modeling to provide complementary results is appealing.

Also of interest in the above analysis are the parameters not included in the model, which can help in identifying actions that are unimportant. Note from Table 2 that the "change order" action variable contributes nothing. In other words, it appears to make no difference, insofar as fitness is concerned, whether agents change posture before or after they shoot at nearby red agents. To some extent, this effect is confounded with the posture main effect: agents shouldn't be in offensive posture to begin with, and so they shouldn't be changing from offensive posture either before or after shooting.

Another variable missing from the model is the "at goal" status variable. Under the conditions of this simulation, virtually the only way for a blue agent to reach the goal is for all the red agents to be dead. In the best rule base we encountered, for example, the average number of red agents alive at the end of 100 time steps is 0.02. Thus, it doesn't matter what a blue agent does or doesn't do by the time he gets to the goal.

Some of the interaction terms in Table 2 simply measure the effect of certain circumstances. For example, consider the SeeBlue$\times$Look interaction term, which is negative (-0.06). Even though it may be best to look when having sensed blue agents nearby, there is less benefit to doing so than looking when blue agents are not nearby.

Some of the estimated coefficients in Table 2 may appear counterintuitive at first. The coefficient for the main effect of sensing red is large and positive, reflecting the fact that rule bases which performed best also sensed red agents frequently, while those performing worst (e.g., if blue agents never moved enough encounter red agents, they never moved enough to reach the goal) sensed red the least.

[Note. The standard errors, t statistics, and p values as give in Table 3 must be taken with a grain of salt. There is no theory formally justifying the modeled relationship (i.e., that the response behaves linearly in the log odds ratios) and the heteroscedastic nature of the data (i.e., the variability in the response increases with the magnitude of the response). Consequently, the usual least squares output as listed in the table should be viewed as an approximation to a more detailed model, whose purpose is to help guide the search of rule base space.]

3.4 A Good Rule Base

Of the rule bases actually simulated, the one having the best fitness (as measured by the average number of blue agents reaching the goal and being alive at the end of 100 time steps) is given in Table 3. Its fitness, averaged over 200 simulated runs, is 4.6 blue agents at the goal, substantially greater than the 3.5 agents to the goal for the rule base determined a priori by the human expert.

Some primary features of this rule base are that blue agents:

(a) are always in defensive posture,

(b) move most of the time, but never when under fire,

(c) shoot whenever they sense red, and

(d) are never under fire by the time they reach the goal.

It is *not* claimed that the rule base in Table 3 maximizes performance over all 10^{35} possible rule bases. Indeed, because only about 1500 different rule bases were actually simulated by the sequential design, many combinations of rules weren't even considered, and the above analysis only summarized effects of actions that were actually taken.

One method to marginally improve the rule base in Table 3 is to do a local search around it. For example, making small changes to the corresponding bit string and simulating would likely allow for a local improvement. If expert opinion were to be used, certain obvious improvements to the above rule base (e.g., agents should always look because there is no penalty for doing so) could also be considered. Still another possibility is to use the best several rule bases from the sequential approach as constituting a first generation for an evolutionary algorithm or other heuristic optimization algorithm.

Included in Table 3 are two columns labeled "Mean" and "SD." The "Mean" column summarizes the average number of times, over 200 replications of the rule base, that each combination of status variables occurred during simulation of the rule base. The "SD" column is the standard deviation associated with the number of times those circumstances occurred. While rules that fire often need not be important (e.g., if most of the time it doesn't matter what blue agents do, but there are crucial moments where it matters greatly), examining the number of times each rule fires is helpful in understanding the dynamics of the simulation. In the rule base summarized

Red	Blue	Fire	Posture	AtGoal	Shoot	Move	ΔPosture	Look	ΔOrder	Mean	SD
0	0	0	0	0	0	0	0	1	1	12.2	2.3
1	0	0	0	0	1	1	0	0	1	0.2	0.7
0	1	0	0	0	0	1	0	1	0	178.2	1.5
1	1	0	0	0	1	1	0	1	0	182.6	45
0	0	1	0	0	0	0	0	1	0	1.1	0.8
1	0	1	0	0	1	0	0	1	1	0	0
0	1	1	0	0	0	0	0	1	1	0.1	0.3
1	1	1	0	0	1	0	0	0	0	5.9	2.3
0	0	0	1	0	0	0	1	0	0	8	0
1	0	0	1	0	1	1	1	0	1	0	0
0	1	0	1	0	0	0	1	1	0	0	0
1	1	0	1	0	1	1	1	1	0	0	0
0	0	1	1	0	0	0	1	0	1	0	0
1	0	1	1	0	1	1	1	1	1	0	0
0	1	1	1	0	0	0	1	1	1	0	0
1	1	1	1	0	1	1	1	0	0	0	0
0	0	0	0	1	0	0	0	1	1	83.6	28.6
1	0	0	0	1	1	0	0	0	1	0.3	1.4
0	1	0	0	1	0	0	0	0	0	65.6	24.7
1	1	0	0	1	1	1	0	0	1	20.1	6.7
0	0	1	0	1	0	0	0	0	0	0	0
1	0	1	0	1	1	1	0	0	0	0	0
0	1	1	0	1	0	1	0	0	1	0	0
1	1	1	0	1	1	0	0	1	0	0	0
0	0	0	1	1	0	1	1	1	0	0	0
1	0	0	1	1	1	0	1	1	1	0	0
0	1	0	1	1	0	1	1	0	1	0	0
1	1	0	1	1	1	1	1	0	1	0	0
0	0	1	1	1	0	0	1	0	1	0	0
1	0	1	1	1	1	0	1	0	1	0	0
0	1	1	1	1	0	1	1	1	0	0	0
1	1	1	1	1	1	0	1	0	1	0	0

Table 3: Best rule base encountered during sequential design.

in Table 3, for example, many combinations of status variables never occur at all.

4 Discussion

4.1 Modeling Benefits

As illustrated in the example, it is possible to extract simple rules from empirical models. Working in the space of odds ratios has advantages relative to working in the space of bit strings or trees, as already noted. Obvious important actions (e.g., agents should move, look, and shoot if sense red agents nearby) can be extracted, together with less obvious ones (e.g., always remain in defensive posture) as well as actions that make no difference (e.g., it doesn't matter what agents do when at the goal). Such basic guidelines can be communicated in a training environment.

If standard optimization techniques were to be used in searching the space of rule bases, the data generated would be useful for regression modeling, in order to better understand which rules matter and which ones don't. Such an effort would involve obtaining results from *all* rule bases involved in the search, not just from the ones giving best results. And, in a general setting, the simulation code must be research friendly insofar as producing output for important variables of interest, such as odds ratios.

We have found that modeling of log odds ratios is also helpful in debugging. By running many rule bases, especially those augmented by partial randomization, results are obtained from regions of the rule base that may not have been fully thought through. Rule bases which lead to outliers from the model are obvious candidates for further inspection. A useful tool in this regard is to save the random number seeds along with other simulation output, thereby allowing results to be reproduced in debug mode, where identifying errors in the code is more straightforward.

All of which is not to say that modeling is the *only* way to extract useful rules and help detect coding errors, of course. But it nicely complements existing methods to those ends.

4.2 Rule Base Fitness

The fitness of a rule base is heavily dependent on the scenario and other parameters. As noted earlier, the act of remaining in defensive posture throughout works well for the current 100 time step limit, but might not be so effective if it were necessary to reach the goal more quickly. Along similar lines, a don't-move-very-often-when-under-fire rule allows for blue agents to improve clustering when attacking red; of course, if red agents had hand grenades, that same clustering could be a poor tactic.

Generally speaking, rule base fitness can also depend on the underlying variability. Consider the case where the fitness function is the probability of the blue team killing all of the red agents. When the blue team is decidedly superior to the red team, it is often to the blue team's advantage to decrease the spread of outcomes. On the other hand, if the blue team were inferior, it might want to increase the spread in order to give it the best chance of winning. In the case at hand, it turns out that good rule bases lead to slightly over half of the blue agents reaching the goal, so that maximizing the average is a reasonable objective.

In many cases, rule base fitness has a multivariate flavor. For example, it may be desirable for the blue team to win the battle as quickly as possible and to minimize its casualties in the process. Such goals may conflict with each other, and definition of rule base fitness must incorporate multiple considerations. In these cases, building models separately for each consideration aids in understanding the big picture.

Lastly relative to the subject of rule base fitness is the issue of co-evolution. Once the blue team establishes tactics, the red team may find it desirable to adjust its own rule base. Such adjustment could be pursued using the modeling process as described here, simply reversing the roles of blue and red. In the extreme, co-evolution has a game-theoretic infinite regress component to it, complicating the optimization.

4.3 Future Efforts

As noted earlier, the sequential design and analysis for JIVES Town data was done subjectively. That is, there was no "cookbook" or recipe that was followed. In an ideal world, sequential design and analysis could be completely automated. If that were the case, users with little knowledge of statistics and experimental design could take full advantage of the methodology. Unfortunately, there aren't any good expert systems around for doing regression modeling on the fly with decent diagnostics (e.g., looking for outliers, and so on). And while assorted subset selection algorithms have been implemented in commercial software, using those algorithms to define rule bases for subsequent simulations is also problematic. Consequently, we have focused on a manual implementation of the approach, postponing automation until a later date.

More immediately, direct comparison can be made between the modeling approach described herein and standard optimization procedures, such as simulated annealing. Although it seems likely that those procedures would find slightly better rule bases (while requiring considerably more code runs in the process), such speculation should be confirmed.

The JIVES Town simulation is simplified in many ways. Such simplifications will be addressed in future versions of the code. At present, there are relatively few status and action variables (e.g., agents move only along a predetermined path, do not communicate with each other, have no memory, and so on), and all such variables are binary. And all agents consult a single rule base, in contrast to the case where different types of agents consult different rule bases. We note that the methodology presented here is extendable to handle a larger number of discrete variables having multiple categories and that odds ratios can be formed for $I \times J$ tables (e.g., Agresti 1990). We are not aware of the use of such ratios in regression models, however, and experience would need to be gained in this regard.

Many methods exist to obtain rule base classifiers from a single data set (see, e.g., Murthy 1998). Bayesian methods for CART (classification and regression trees; see, e.g., Chipman, George, and McCulloch 1998; Denison, Mallick, and Smith 1998) are also used for such optimization. It seems possible that the techniques used there to search spaces of trees could be adapted to the JIVES Town problem. An issue to be overcome is that the problem here is "backwards" from the usual CART approach. There, the idea is to start with the single data set, presumably evolving from a single rule base, and search the space of trees to find the one that gives the best fit to the data. When done, that tree can be viewed as an estimate of the underlying rule base.

For the JIVES Town problem, multiple data sets are generated from known rule bases and the the approach is similar to so-called data farming. That is, data are "grown" from several rule bases. Certain optimization procedures, such as simulated annealing, search the space in their own way, and could possibly gain from methods underlying CART searches.

The biggest obstacle concerns the dimension of rule base space. Even for the simplified version of JIVES Town, there are roughly 10^{35} allowable rule bases. It is not hard to imagine problems where the space to be searched is so large as to strain comprehension. The curse of dimensionality could affect approaches to optimization that work in bit string space more adversely that it affects regression modeling, however. In that bit string space is of higher dimension and rule base fitness behaves discontinuously as a function of it, modeling of odds ratios may prove to be a more practical way of extracting useful tactics.

References

Agresti, A. (1990), *Categorical Data Analysis*, New York: John Wiley.

Bishop, Y. M., Fienberg, S. E., and Holland, P. W. (1975), *Discrete Multivariate Analysis.* Cambridge: MIT Press.

"CASTFOREM (Combined Arms and Support Task Force Evaluation Model) Update: Scenario Writers Guide," Army TRADOC Analysis Center technical report TRAC-WSMR-TD-99-004. 1999.

Chipman, H. A., George, E. I., and McCulloch, R. E. (1998), "Bayesian CART Model Search" *Journal of the American Statistical Association* **93** 935-947.

Deadman, P. J. (1999), "Modelling Individual Behaviour and Group Performance in an Intelligent Agent-Based Simulation of the Tragedy of the Commons," *Journal of Environmental Management* **56** 159-172.

Denison, D. G., Mallick, B. K., and Smith, A. F. M. (1998), "A Bayesian CART Algorithm," *Biometrika* **85** 363-377.

Fogerty, T. C. and Bull, L. (1995), "Optimising Individual Control Rules and Multiple Communicating Rule-Based Control Systems with Parallel Distributed Genetic Algorithms," *IEE Proceedings - Control Theory and Applications* **142** 211-215.

Ilachinski, A. (1997)," Irreducible Semi-Autonomous Adaptive Combat (ISAAC): An Artificial-Life Approach to Land Warfare," Center for Naval Analyses Research Memorandum CRM 97-61.

Jaffe, K., Issa, S., Daniels, E., and Haile, D. (1997), "Dynamics of the Emergence of Genetic Resistance to Biocides among Asexual and Sexual Organisms," *Journal of Theoretical Biology* **188** 289-299.

Murthy, S. K. (1998), "Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey," *Data Mining and Knowledge Discovery* **2** 345-389.

Nagel, K., Rickert, M., and Barrett, C. L. (1997), "Large Scale Traffic Simulations," *Lecture Notes in Computer Science* **1215** 380-402.

Samuels, M. L. (1993), "Simpson's Paradox and Related Phenomena" *Journal of the American Statistical Association* **88** 81-88.

"TAC THUNDER Management Summary," CACI Defense Technical Information Center Report AD-B175 308/XAG (1987).

Upton, S. C., Michelsen, R. E., Powell, D. R., Holland, J. V., Thompson, D., and R. R. Picard (1998), "Generative Analysis: Proof-of-Concept Final Report," Los Alamos National Laboratory technical report LA-UR-98-5996.